AFRL-AFOSR-VA-TR-2016-0350

Concurrency Attacks and Defenses

Junfeng Yang
THE TRUSTEES OF COLUMBIA UNIVERSITY IN THE CITY OF NEW YORK

10/04/2016
Final Report

DISTRIBUTION A: Distribution approved for public release.

Air Force Research Laboratory
AF Office Of Scientific Research (AFOSR)/ RTA2
Arlington, Virginia 22203
Air Force Materiel Command

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*

2. REPORT TYPE
Final

3. DATES COVERED *(From - To)*
7/1/12-6/30/16

4. TITLE AND SUBTITLE
Concurrency Attacks and Defenses

5a. CONTRACT NUMBER
FA9550-12-1-0346

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Associate Professor Junfeng Yang

5d. PROJECT NUMBER
GG006956

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
The Trustees of Columbia University in the City of New York DUNS 049179401
Sponsored Projects Administration, Morningside
615 West 131st Street, Room 254, Mail Code 8725
New York, NY 10027-7922

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
USAF, AFRL DUNS 143574726
AF OFFICE OF SCIENTIFIC RESEARCH
875 NORTH RANDOLPH STREET, RM 3112
ARLINGTON VA 22203

10. SPONSOR/MONITOR'S ACRONYM(S)
AFOSR

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTUION A

13. SUPPLEMENTARY NOTES

14. ABSTRACT
Multithreaded programs are getting increasingly pervasive and critical. Unfortunately, they remain extremely difficult to write. This difficulty has led to many subtle but serious concurrency vulnerabilities such as race conditions in real-world multithreaded programs. Just as vulnerabilities in sequential programs can lead to security exploits, concurrency vulnerabilities can also be exploited by attackers to gain privilege, steal information, inject arbitrary code, etc. The objective of this project is to take a holistic approach to creating novel program analysis/protection techniques and a system called DASH to secure multithreaded programs and harden traditional defense techniques in a concurrency environment.

15. SUBJECT TERMS
Concurrency attacks, Multithreading, Vulnerability, Security, Software Hardening

16. SECURITY CLASSIFICATION OF: a. REPORT	b. ABSTRACT	c. THIS PAGE	17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
U	U	U	SAR		19b. TELEPHONE NUMBER *(Include area code)*

Reset

Standard Form 298 (Rev. 8/98)
Prescribed by ANSI Std. Z39.18
Adobe Professional 7.0

INSTRUCTIONS FOR COMPLETING SF 298

1. REPORT DATE. Full publication date, including day, month, if available. Must cite at least the year and be Year 2000 compliant, e.g. 30-06-1998; xx-06-1998; xx-xx-1998.

2. REPORT TYPE. State the type of report, such as final, technical, interim, memorandum, master's thesis, progress, quarterly, research, special, group study, etc.

3. DATES COVERED. Indicate the time during which the work was performed and the report was written, e.g., Jun 1997 - Jun 1998; 1-10 Jun 1996; May - Nov 1998; Nov 1998.

4. TITLE. Enter title and subtitle with volume number and part number, if applicable. On classified documents, enter the title classification in parentheses.

5a. CONTRACT NUMBER. Enter all contract numbers as they appear in the report, e.g. F33615-86-C-5169.

5b. GRANT NUMBER. Enter all grant numbers as they appear in the report, e.g. AFOSR-82-1234.

5c. PROGRAM ELEMENT NUMBER. Enter all program element numbers as they appear in the report, e.g. 61101A.

5d. PROJECT NUMBER. Enter all project numbers as they appear in the report, e.g. 1F665702D1257; ILIR.

5e. TASK NUMBER. Enter all task numbers as they appear in the report, e.g. 05; RF0330201; T4112.

5f. WORK UNIT NUMBER. Enter all work unit numbers as they appear in the report, e.g. 001; AFAPL30480105.

6. AUTHOR(S). Enter name(s) of person(s) responsible for writing the report, performing the research, or credited with the content of the report. The form of entry is the last name, first name, middle initial, and additional qualifiers separated by commas, e.g. Smith, Richard, J, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES). Self-explanatory.

8. PERFORMING ORGANIZATION REPORT NUMBER. Enter all unique alphanumeric report numbers assigned by the performing organization, e.g. BRL-1234; AFWL-TR-85-4017-Vol-21-PT-2.

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES). Enter the name and address of the organization(s) financially responsible for and monitoring the work.

10. SPONSOR/MONITOR'S ACRONYM(S). Enter, if available, e.g. BRL, ARDEC, NADC.

11. SPONSOR/MONITOR'S REPORT NUMBER(S). Enter report number as assigned by the sponsoring/ monitoring agency, if available, e.g. BRL-TR-829; -215.

12. DISTRIBUTION/AVAILABILITY STATEMENT. Use agency-mandated availability statements to indicate the public availability or distribution limitations of the report. If additional limitations/ restrictions or special markings are indicated, follow agency authorization procedures, e.g. RD/FRD, PROPIN, ITAR, etc. Include copyright information.

13. SUPPLEMENTARY NOTES. Enter information not included elsewhere such as: prepared in cooperation with; translation of; report supersedes; old edition number, etc.

14. ABSTRACT. A brief (approximately 200 words) factual summary of the most significant information.

15. SUBJECT TERMS. Key words or phrases identifying major concepts in the report.

16. SECURITY CLASSIFICATION. Enter security classification in accordance with security classification regulations, e.g. U, C, S, etc. If this form contains classified information, stamp classification level on the top and bottom of this page.

17. LIMITATION OF ABSTRACT. This block must be completed to assign a distribution limitation to the abstract. Enter UU (Unclassified Unlimited) or SAR (Same as Report). An entry in this block is necessary if the abstract is to be limited.

Final Report: Project Concurrency Attacks & Defenses

1. PI and contact information

PI: Junfeng Yang
Contact Information:

Junfeng Yang
Department of Computer Science
M.C. 0401
Columbia University
1214 Amsterdam Avenue
New York, NY 10027-7003
Tel.: 212-939-7012
Fax: 212-666-0140
Email: junfeng@cs.columbia.edu

2. Research Objectives

Multithreaded programs are getting increasingly pervasive and critical. Unfortunately, they remain extremely difficult to write. This difficulty has led to many subtle but serious concurrency vulnerabilities such as race conditions in real-world multithreaded programs. Just as vulnerabilities in sequential programs can lead to security exploits, concurrency vulnerabilities can also be exploited by attackers to gain privilege, steal information, inject arbitrary code, etc. Concurrency attacks targeting these vulnerabilities are impending (see CVE http://www.cvedetails.com/vulnerability-list/cweid-362/vulnerabilities.html), yet few existing defense techniques can deal with concurrency vulnerabilities. In fact, many of the traditional defense techniques are rendered unsafe by concurrency vulnerabilities.

The objective of this project is to take a holistic approach to creating novel program analysis/protection techniques and a system called DASH to secure multithreaded programs and harden traditional defense techniques in a concurrency environment. We do so by selectively combining static and dynamic techniques, thus getting the best of both worlds. We anticipate numerous contributions from this project; the main ones are: (1) a thorough understanding of concurrency attacks and their implications to traditional defense techniques;

(2) accurate and effective techniques to detect, avoid, and survive concurrency vulnerabilities; and (3) hardening of traditional defense techniques for multithreaded programs. The greatest impact of our project is a novel approach and the DASH system for improving software security and reliability, thus greatly benefiting the Nation's cyber security. DASH can also be used for offense: the Military can gain new competitive means in cyber warfare by running DASH to identify concurrency vulnerabilities in the infrastructure of hostile nations.

3. Brief Description of the proposed efforts

Two trends have caused multithreaded programs to become pervasive and critical. The first is a hardware trend: the rise of multicore computing. For years, sequential code enjoyed automatic speedup as computer architects steadily made single-core multiprocessors faster. Recently, however, this "free lunch is over": power and wire-delay constraints have forced microprocessors into multicore designs, and adding more cores does not automatically speed up sequential code. Thus, developers, including those working for various government agencies, are writing more and more multithreaded code.

The second trend is a software one: the coming storm of cloud computing. More and more services, including many traditionally offered on desktops (*e.g.*, word processing), are now served from distributed "clouds" of servers to meet the current computing demands for high scalability, always-on availability, everywhere connectivity, and desirable consistency. These services are also getting ever richer and more powerful—and thus computation and data intensive. To cope with this massive workload, practically all services today employ threads to increase performance.

Unfortunately, despite our increasing reliance on multithreaded programs, they remain extremely difficult to write. This difficulty has led to many subtle but serious *concurrency vulnerabilities* such as race conditions in real-world multithreaded programs. Multithreaded programs are the most widespread parallel programs, yet many luminaries in computing consider parallel programming one of the top challenges facing computer science. As John Hennessy once put: *"when we start talking about parallelism and ease of use of truly parallel computers, we're talking about a problem that's as hard as any that computer science has faced."* Just as vulnerabilities in sequential programs can lead to security exploits, concurrency vulnerabilities can similarly compromise security and lead to what we call *concurrency attacks*. Our recent study of real concurrency vulnerabilities shows that these vulnerabilities are very dangerous: they allow attackers to corrupt arbitrary program data, inject malicious code, and escalate privileges. Worse, in addition to being directly exploited by attackers, concurrency vulnerabilities also compromise key defense techniques we used to trust. For instance, consider an information flow tracking mechanism that tracks whether each piece of program data is classified or not using a metadata tag. An attacker may exploit a race condition on program

data to make the data and the tag inconsistent, thus evading the information flow tracking mechanism.

We believe concurrency attacks are an imminent threat to the Nation's cyber security: they are becoming a major form of future cyber-attacks. Unfortunately, we currently lack a thorough understanding of concurrency attacks. Nor do we have automated and effective techniques to detect, avoid, or survive concurrency vulnerabilities. Despite repeated efforts, precisely and comprehensively analyzing multithreaded programs has been an open challenge for at least three decades.

The key reason that multithreaded programs are so difficult to analyze is that each run of a multithreaded program may interleave the threads differently. For a typical multithreaded program, the number of these thread interleavings, or *schedules*, is enormous—asymptotically exponential in the program size. Existing methods to analyze multithreaded programs are either static (compile-time) or dynamic (runtime), yet both have difficulties analyzing this enormous number of schedules. Specifically, static techniques can analyze all statements that a compiler can see, but it is not good at reasoning about runtime behaviors such as all possible schedules that may occur. Thus, it has to over-approximate these schedules, often resulting in poor precision and tons of false positives when applied to concurrency vulnerability detection. Dynamic techniques can precisely reason about the schedules executed at runtime. However, they can analyze only the schedules occurred, and they rarely cover more than a tiny fraction of all possible schedules. Thus, the next execution of a multithreaded program may well use an unchecked schedule and run into a concurrency vulnerability.

The objective of this proposal is to take a holistic approach to creating novel program analysis/protection techniques and the DASH system to effectively detect, avoid, and survive concurrency vulnerabilities and harden traditional defense techniques in a concurrency environment. We address this open challenge of creating secure and reliable multithreaded programs by selectively combining static and dynamic techniques, thus getting the best of both worlds. Specifically, we guide static analysis using real schedules observed at runtime. By targeting static analysis toward only the schedules that matter, we drastically improve its precision. We then protect the execution of multithreaded programs by dynamically enforcing the schedules that are thoroughly checked and deemed free of concurrency vulnerabilities. The feedback loop formed by integrating static and dynamic analysis positively enhance each other, resulting in more secure and reliable multithreaded programs.
To fully evaluate our approach, we plan to go "from soup to nuts" in designing and building a prototype system of DASH and applying it to real multithreaded programs. Besides the DASH system and the novel approach of combining static and dynamic techniques to harden multithreaded programs, we anticipate five additional contributions within the proposed research:

• A *thorough study* of concurrency attacks and their implications to traditional defense techniques;

• *Schedule-guided detection*, a static analysis technique to accurately and effectively detect concurrency vulnerabilities;

• *Schedule enforcement*, a software protection technique to enforce the schedules that are well checked and deemed correct, thus avoiding potential concurrency vulnerabilities in unchecked schedules;

• *Schedule diversification*, a software diversification technique to survive previously unknown concurrency vulnerabilities; and

• *Hardening of traditional defense techniques* such as taint tracking and information flow tracking for multithreaded programs.

Applications of the Proposed Research. The greatest impact of our project is a novel approach and new, effective systems and technologies to improve software security and reliability, greatly benefiting the Nation's cyber security. I plan to leverage my long-term connections with Microsoft, the biggest software company, and Coverity, a top software quality assurance startup, to make the technologies developed within this effort available to the public (see letters of support for technology transfer opportunities). We envision numerous applications of the proposed research. The main ones include:

1. The thorough understanding of concurrency vulnerabilities we develop helps other researchers and practitioners to come up with new defense techniques.

2. Developers of multithreaded programs can use DASH to accurately detect concurrency vulnerabilities in their code. By fixing these concurrency vulnerabilities, they create more secure and reliable multithreaded programs that can both efficiently use the power of multicore hardware and effectively fend off concurrency attacks.

3. Users of multithreaded programs can use DASH to avoid or survive concurrency vulnerabilities at runtime, further strengthening the Nation's cyber infrastructure and increasing its resilience against concurrency attacks.

4. Builders of traditional defense techniques such as information flow tracking can use DASH to harden their techniques and achieve the same safety guarantees for both sequential and multithreaded programs.

5. Alternatively, DASH can be used for offense. For instance, the Military can gain new competitive means in cyber warfare by running DASH to identify concurrency vulnerabilities in the infrastructure of hostile nations. Furthermore, although we focus on multithreaded programs in this effort, many proposed techniques benefit concurrent programs written in other programming models, such as MPI and OpenMP.

4. Results from the project

Since the beginning of the project, we have made tremendous progress developing four components of the project, including the *thorough study* of concurrency attacks and their implications to traditional defense techniques; *schedule-guided detection*, a static analysis technique to accurately and effectively detect concurrency vulnerabilities; *schedule enforcement and diversification*, software protection techniques to enforce the schedules that are well checked and deemed correct, thus avoiding potential concurrency vulnerabilities in unchecked schedules; and *Hardening of traditional defense techniques* such as taint tracking and information flow tracking for multithreaded programs. Our results have led to total 15 publications at the best venues, 26 invited talks at many universities and research institutes, and several releases of open source software.

Relevant Publications (total 15):

1. Eric Koskinen and Junfeng Yang. ***Reducing crash recoverability to reachability.*** In Proceedings of the 39th Annual Symposium on Principles of Programming Languages (POPL ’16), January 2016. (acceptance rate: 23.3%, 59/253)

2. Heming Cui, Rui Gu, Cheng Liu, Tianyu Chen, and Junfeng Yang. ***Paxos made transparent.*** In Proceedings of the 25th ACM Symposium on Operating Systems Principles (SOSP ’15), October 2015. (acceptance rate: 16.1%, 30/186)

3. Xinhao Yuan, David Williams-King, Junfeng Yang, and Simha Sethumadhavan. ***Making lock-free data structures verifiable with artificial transactions.*** In Eighth Workshop on Programming Languages and Operating Systems (PLOS ’15), October 2015.

4. Heming Cui, Rui Gu, Cheng Liu, and Junfeng Yang. ***Repframe: An efficient and transparent framework for dynamic program analysis.*** In Proceedings of 6th Asia-Pacific Workshop on Systems (APSys ’15), July 2015.

5. Yang Tang and Junfeng Yang. ***Secure deduplication of general computations.*** In Proceedings of the USENIX Annual Technical Conference (USENIX ATC ’15), 2015. (acceptance rate: 15.8%, 35/221)

6. Suzanna Schmeelk, Junfeng Yang, and Alfred Aho. ***Android malware static analysis techniques.*** In The 10th Annual Cyber and Information Security Research (CISR) Conference, 2015.

7. Yinzhi Cao and Junfeng Yang. ***Towards making systems forget with machine unlearning. In Proceedings of the 2015 IEEE Symposium on Security and Privacy***. In Proceedings of the 2015 IEEE Symposium on Security and Privacy (S&P ’15), 2015. (acceptance rate: 13.5%, 55/407)

8. Gang Hu, Xinhao Yuan, Yang Tang, and Junfeng Yang. ***Efficiently, effectively detecting mobile app bugs with AppDoctor.*** In Proceedings of the 2014 ACM European Conference on Computer Systems (EUROSYS ’14), April 2014. (acceptance rate: 18.4%, 27/147)

9. Junfeng Yang, Heming Cui, Jingyue Wu, Yang Tang, and Gang Hu. ***Determinism is not enough: Making parallel programs reliable with stable multithreading.*** Communications of the ACM, 2014. (invited)

10. Heming Cui, Jiri Simsa, Yi-Hong Lin, Hao Li, Ben Blum, Xinan Xu, Junfeng Yang, Garth A. Gibson, and Randal E. Bryant. ***Parrot: a practical runtime for deterministic, stable, and reliable threads.*** In Proceedings of the 24th ACM Symposium on Operating Systems Principles (SOSP ’13), November 2013. (acceptance rate: 18.8%, 30/160)

11. Jingyue Wu, Gang Hu, Yang Tang, and Junfeng Yang. ***Effective dynamic detection of alias analysis errors***. In Proceedings of the Ninth Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC-FSE ’13), August 2013. (acceptance rate: 20.3%, 51/251)

12. Junfeng Yang, Heming Cui, and Jingyue Wu. ***Determinism is overrated: What really makes multithreaded programs hard to get right and what can be done about it?*** In the Fifth USENIX Workshop on Hot Topics in Parallelism (HOTPAR ’13), June 2013.

13. Heming Cui, Gang Hu, Jingyue Wu, and Junfeng Yang. ***Verifying systems rules using rule-directed symbolic execution***. In Eighteenth International Conference on Architecture Support for Programming Languages and Operating Systems (ASPLOS ’13), March 2013. (acceptance rate: 23.1%, 44/191)

    Also appeared as journal publication:
    Heming Cui, Gang Hu, Jingyue Wu, and Junfeng Yang. Verifying systems rules using rule-directed symbolic execution. SIGARCH Comput. Archit. News, 41(1):329–432, 2013.

14. Jingyue Wu, Yang Tang, Gang Hu, Heming Cui, and Junfeng Yang. ***Sound and precise analysis of parallel programs through schedule specialization***. In Proceedings of the ACM SIGPLAN 2012 Conference on Programming Language Design and Implementation (PLDI ’12), pages 205–216, June 2012. (acceptance rate: 18.8%, 48/255)

    Also appeared as journal publication:
    Jingyue Wu, Yang Tang, Gang Hu, Heming Cui, and Junfeng Yang. Sound and precise analysis of parallel programs through schedule specialization. SIGPLAN Not., 47(6):205–216, June 2012.

15. Junfeng Yang, Ang Cui, Sal Stolfo, and Simha Sethumadhavan. ***Concurrency attacks. In the Fourth USENIX Workshop on Hot Topics in Parallelism*** (HOTPAR ’12), June 2012.

Relevant Invited Talks (total 26):

1. 7/2016 “Need for Speed: Software Tools Edition.” Microsoft Research Faculty Summit. Host: Suman Nath
2. 3/2016 “Build Performant Apps: Metrics, Common Issues, and Best Practices.” Droidcon San Francisco. Host: Apps4all, Touchlab
3. 12/2015 “Build Fluid Apps with Android Profiling Tools.” AnDevCon Santa Clara. Host: BZ Media
4. 7/2015 “Build Fluid Apps with Android Profiling Tools.” AnDevCon Boston. Host: BZ Media
5. 4/2014 “Determinism Is Not Enough: Making Parallel Programs Reliable with Stable Multithreading.” Princeton University. Host: Michael Freedman
6. 4/2014 “Determinism Is Not Enough: Making Parallel Programs Reliable with Stable Multithreading.” University of Washington. Host: Tom Anderson
7. 02/2014 “Determinism Is Not Enough: Making Parallel Programs Reliable with Stable Multithreading.” UCLA. Host: Todd Millstein
8. 12/2013 “Determinism Is Not Enough: Making Parallel Programs Reliable with Stable Multithreading.” UT Austin. Host: Emmett Witchel
9. 11/2013 “Parrot: A Practical Runtime for Deterministic, Stable, and Reliable Threads.” Princeton University. Host: Michael Freedman
10. 11/2013 “Determinism Is Not Enough: Making Parallel Programs Reliable with Stable Multithreading.” UCSD. Host: Yuanyuan Zhou
11. 11/2013 “Determinism Is Not Enough: Making Parallel Programs Reliable with Stable Multithreading.” Stanford University. Host: Alex Aiken
12. 10/2013 “Determinism Is Not Enough: Making Parallel Programs Reliable with Stable Multithreading.” Cornell University. Host: Andrew Myers
13. 08/2013 “Determinism Is Not Enough: Making Parallel Programs Reliable with Stable Multithreading.” Microsoft Research Asia. Host: Lintao Zhang

14. 08/2013 “Determinism Is Not Enough: Making Parallel Programs Reliable with Stable Multithreading.” Beijing University. Host: Yao Guo

15. 08/2013 “Determinism Is Not Enough: Making Parallel Programs Reliable with Stable Multithreading.” Shanghai Jiaotong University. Host: Haibo Chen

16. 07/2013 “Effectively Model Check Real-World Distributed Systems.” National University of Singapore. Host: Jin Song Dong

17. 06/2013 “How Useful Is Determinism for Reliability?” Invited panel “Determinism: Blessing or Curse” at Fifth USENIX Workshop on Hot Topics in Parallelism. Host: Emery Berger and Kim Hazelwood

18. 04/2013 “Effectively Model Check Real-World Distributed Systems.” Rutgers. Host: Santosh Nagarakatte

19. 12/2012 “Effectively Model Check Real-World Distributed Systems.” CMU. Host: Garth Gibson

20. 10/2012 “Pervasive Detection of Process Races in Deployed Systems.” University of Southern California. Host: Minlan Yu

21. 06/2012 “Improving the Reliability and Security of Parallel Programs.” Tsinghua University. Host: Wenguang Chen

22. 06/2012 “Improving the Reliability and Security of Parallel Programs.” Beihang University. Host: Chunming Hu

23. 06/2012 “Improving the Reliability and Security of Parallel Programs.” Beijing University. Host: Yao Guo

24. 06/2012 “Efficiently and Stably Making Threads Deterministic.” Invited talk at 4th International Workshop on Practical Synthesis (co-located with PLDI). Host: Martin Vechev and Eran Yahav

25. 12/2011 “Efficiently and Stably Making Threads Deterministic.” Microsoft Research. Host: Madan Musuvathi

26. 11/2011 “Efficiently and Stably Making Threads Deterministic.” Telefonica Research at Spain. Host: Michael Sirivianos

AFOSR Deliverables Submission Survey

Response ID:6989 Data

1.

Report Type

Final Report

Primary Contact Email
Contact email if there is a problem with the report.

junfeng@cs.columbia.edu

Primary Contact Phone Number
Contact phone number if there is a problem with the report

212-939-7012

Organization / Institution name

Columbia University

Grant/Contract Title
The full title of the funded effort.

Concurrency Attacks and Defenses

Grant/Contract Number
AFOSR assigned control number. It must begin with "FA9550" or "F49620" or "FA2386".

FA9550-12-1-0346

Principal Investigator Name
The full name of the principal investigator on the grant or contract.

Junfeng Yang

Program Officer
The AFOSR Program Officer currently assigned to the award

Ria Miranda

Reporting Period Start Date

07/01/2012

Reporting Period End Date

06/30/2016

Abstract

Multithreaded programs are getting increasingly pervasive and critical. Unfortunately, they remain extremely difficult to write. This difficulty has led to many subtle but serious concurrency vulnerabilities such as race conditions in real-world multithreaded programs. Just as vulnerabilities in sequential programs can lead to security exploits, concurrency vulnerabilities can also be exploited by attackers to gain privilege, steal information, inject arbitrary code, etc. Concurrency attacks targeting these vulnerabilities are impending (see CVE http://www.cvedetails.com/vulnerability-list/cweid-362/vulnerabilities.html), yet few existing defense techniques can deal with concurrency vulnerabilities. In fact, many of the traditional defense techniques are rendered unsafe by concurrency vulnerabilities.

The objective of this project is to take a holistic approach to creating novel program analysis/protection techniques and a system called DASH to secure multithreaded programs and harden traditional defense techniques in a concurrency environment. We do so by selectively combining static and dynamic techniques, thus getting the best of both worlds. We anticipate numerous contributions from this project; the main ones are: (1) a thorough understanding of concurrency attacks and their implications to traditional

defense techniques; (2) accurate and effective techniques to detect, avoid, and survive concurrency vulnerabilities; and (3) hardening of traditional defense techniques for multithreaded programs. The greatest impact of our project is a novel approach and the DASH system for improving software security and reliability, thus greatly benefiting the Nation's cyber security. DASH can also be used for offense: the Military can gain new competitive means in cyber warfare by running DASH to identify concurrency vulnerabilities in the infrastructure of hostile nations.

Distribution Statement

This is block 12 on the SF298 form.

Distribution A - Approved for Public Release

Explanation for Distribution Statement

If this is not approved for public release, please provide a short explanation. E.g., contains proprietary information.

SF298 Form

Please attach your SF298 form. A blank SF298 can be found here. Please do not password protect or secure the PDF
The maximum file size for an SF298 is 50MB.

SF298_Yang_FA9550-12-1-0346.pdf

Upload the Report Document. File must be a PDF. Please do not password protect or secure the PDF . The maximum file size for the Report Document is 50MB.

Final.pdf

Upload a Report Document, if any. The maximum file size for the Report Document is 50MB.

Archival Publications (published) during reporting period:

Please refer to the report.

New discoveries, inventions, or patent disclosures:
Do you have any discoveries, inventions, or patent disclosures to report for this period?

Yes

Please describe and include any notable dates

Please refer to the report.

Do you plan to pursue a claim for personal or organizational intellectual property?

No

Changes in research objectives (if any):

Change in AFOSR Program Officer, if any:

Extensions granted or milestones slipped, if any:

AFOSR LRIR Number

LRIR Title

Reporting Period

Laboratory Task Manager

Program Officer

Research Objectives

Technical Summary

Funding Summary by Cost Category (by FY, $K)

	Starting FY	FY+1	FY+2
Salary			
Equipment/Facilities			
Supplies			
Total			

Report Document

Report Document - Text Analysis

Report Document - Text Analysis

Appendix Documents

2. Thank You

E-mail user

Sep 28, 2016 18:09:13 Success: Email Sent to: junfeng@cs.columbia.edu